AS3 Transition Library v. 2.4.0
Manual

License: Free

Note: This software is provided AS IS no warranty is given as to the functionality or purpose of this software.

General

This library is implemented in Actionscript 3 and is suitable for Flash CS3/CS4/CS5 (Flash Player 9+). It is provided in .swc format which you can include in your .fla file through publish/actionscript settings menu. In Flex you can just include the path.

Dependencies

This library uses PaperVision 3D engine and TweenMax engine. These are included in the .swc file so you don't have to include them separately. The library uses also the AS3 Page Flipper class modified to include additional effects (included in .swc) and the AS3 Rippler Class also modified (also included in .swc). These classes are free and open-source software.

API Parameters

To use the transitions simply include the following code in your source files:
`import com.nikos.mytransitions.*;`
or
`import com.nikos.mytransitions.FadeTiles;` etc.. for specific transitions

Each effect class has 4 static public parameters which define certain aspects of the effects These are common to all classes but they apply to each class alone.

- **useFrames** boolean param (default **true**), indicates whether the duration provided indicates frames or seconds
- **persist** boolean param (default **true**), indicates whether the final image will be visible and persist after the end of the transition
- **dispatch** boolean param (default **true**), indicates whether the class will dispatch events when the transition is over. The event is `Transition.eventType` this is what you should listen to.
- **startNow** boolean param (default **true**), indicates whether the transition should start immediately or on the next ENTER_FRAME event

These parameters are static thus the same for every instance of the same transition class but different for instances of different transition classes.

Also 5 static global public params exist which apply to all effects these are:

- **useFramesGlobal** boolean param (default **true**), indicates whether the duration provided indicates frames or seconds
- **persistGlobal** boolean param (default **true**), indicates whether the final image will be visible and persist after the end of the transition
- **dispatchGlobal** boolean param (default **true**), indicates whether the class will dispatch events when the transition is over. The event is `Transition.eventType` this is what you should listen to.
- **startNowGlobal** boolean param (default **true**), indicates whether the transition should start immediately or on the next ENTER_FRAME event
- **useGlobal** boolean param (default **false**) indicates whether the local or global params should be used. Global params apply to all instances of all classes whereas local params apply to all instances of each specific class alone

All transition classes are subclasses of the Transition Class which u can use as a general substitute for generic transitions.
Eg:

```
FadeTiles.useFrames=false;  // use seconds for all FadeTiles transitions
```

or

```
Transition.useGlobal=true;
Transition.useFramesGlobal=false;  // use seconds for all transitions
```

Public Methods

Each Transition Effect class provides 2 public methods: the `doit` method and the `kill` method. The `doit` method does the transition and accepts an object as parameter which defines the parameters for the specific transition effect (see the documentation for each effect below). The `kill` method accepts no parameters and stops the transition that has started, before the duration is over (forcing it to completion). The `kill` method must only be called after the `doit` method has been called and only if you want to stop immediately the transition that has started.

Calling the `kill` method if the `doit` method is not active does nothing. The `kill` method actually forces the transition to end before the whole duration is over. As such the *Transition Complete event* will be fired (if the `dispatch` parameter is true and there is a listener for it).

Transition Effects

- **FadeTiles (fades in the tiles in the specified ordering)**
- **FlipTiles (flips tiles horizontally or vertically 3D in specified ordering)**
- **ScaleTiles (scales ie grows or shrinks tiles and rotates in specified ordering)**
- **Slices (horizontal, vertical or diagonal slices in specified ordering)**
- **Cubes (rotates horizontally or vertically sliced cubelets in specified direction and ordering)**
- **GradientMasks (sliced gradient alpha masks horizontal or vertical in specified ordering)**
- **Fly (moves new image as if it flies above from specified point)**
- **Clock (clock-like transition with specified number of segments)**
- **FlipPage (flips the image as if it is a page horizontally vertically or diagonally)**
- **FlipPage3D (flips the image in 3D style by bending or twisting)**
- **DreamWave (distorts image in a dream like waving manner)**
- **Pixelate (scales image to pixel size then to next image reversely)**
- **GradientSweep (sweeps image with an alpha gradient mask from specified point)**
- **Ripples (fades in image like it is reflected on rippled water)**
- **Blinds (this is the well known blinds transition only more extended to include more options)**
- **Blur (blurs in the target)**
- **Dissolve (dissolves the image)**
- **Masks (reveals the image using masks)**
- **JigsawPuzzle (decomposes and recomposes images like jigsaw puzzles)**
- **Fold (3D fold-like transition)**
- **Shuffle (3D card shuffle transition)**

Orderings

- **"diagonal-top-left"**
- **"diagonal-top-right"**
- **"diagonal-bottom-left"**
- **"diagonal-bottom-right"**
- **"rows"**
- **"rows-reverse"**
- **"columns"**
- **"columns-reverse"**
- **"checkerboard"**
- **"rows-first"**
- **"rows-first-reverse"**
- **"columns-first"**
- **"columns-first-reverse"**
- **"spiral-top-left"**
- **"spiral-top-right"**
- **"spiral-bottom-left"**
- **"spiral-bottom-right"**
- **"spiral-top-left-reverse"**
- **"spiral-top-right-reverse"**
- **"spiral-bottom-left-reverse"**
- **"spiral-bottom-right-reverse"**
- **"random"**
- **"up-down"**
- **"up-down-reverse"**
- **"left-right"**
- **"left-right-reverse"**

Note: Not all ordering functions apply to all transitions but most are

Overlap Parameter
The Transitions that have an overlap parameter (most of them) is defined as the percentage of overlap between the start of the previous tile animation inside the transition to the start of the next tile animation in the same transition. Thus a value of 0 indicates no overlap (ie next tile starts after previous finishes) and a value of 1 indicates full overlap (all tiles start simultaneously). The values in-between indicate partial overlap. Overlap Parameter takes values 0 – 1 inclusive.

Easing
Each Transition Effect demands an easing function. U can use any easing function u like eg from `fl.transitions.easing` or any other, just pass the function name as transition easing parameter.

Duration
Each Transition Effect demands a duration parameter which defines the duration (in seconds or frames depending on the useFrames and useFramesGlobal parameter) of the whole transition and of the transition only.

Targets
All transitions demand 2 targets (ie display objects) a **fromTarget** and a **toTarget**. More details in the Documentation for each Transition.

Transition Classes Documentation

Dissolve

Description:
Dissolves the target using specified easing

Use example:
```
var foo=new Dissolve();
addChild(foo);
foo.doit({
          toTarget:pic,
          fromTarget:pic2,
          duration:40,
          easing:None.easeInOut});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

Masks

Description:
Reveals the target using masks

Use example:
```
var foo=new Masks();
addChild(foo);
foo.doit({
          toTarget:pic,
          fromTarget:pic2,
          duration:40,
          easing:None.easeInOut});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

Blur

Description:
Blurs the target using specified easing

Use example:
```
var foo=new Blur();
addChild(foo);
foo.doit({
          toTarget:pic,
          fromTarget:pic2,
          duration:40,
          easing:None.easeInOut});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

JigsawPuzzle

Description:
Decomposes and recomposes targets like a jigsaw puzzle

Use example:
```
var foo=new JigsawPuzzle();
addChild(foo);
foo.doit({
          toTarget:pic,
          fromTarget:pic2,
          duration:40,
          easing:None.easeInOut,
          mode:”random-move”,
          overlap:0.92,
          ordering:"random"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on

useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`ordering`: **String**, an ordering function to apply to the tiles, see Orderings (default "random")
`overlap`: **Number (0-1)**, the percentage of overlap between tiles, 0 is no overlap, 1 full overlap, 0.5 half overlap etc.. (default 0.8)
`mode`: **String**, the mode of the puzzle pieces to follow.. (default "random-move") values are:

- "random-move"
- "center-move"
- "swap"
- "rotate"
- "scale"

Fold

Description:
Folds 3D like cards folding

Use example:

```
var foo=new Fold();
addChild(foo);
foo.doit({
          toTarget:pic,
          fromTarget:pic2,
          duration:40,
          easing:None.easeInOut,
          mode:"left"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`mode`: **String**, the direction of folding.. (default "left") values are:

- "left"
- "right"

Shuffle

Description:
3D Shuffle like card shuffling

Use example:
```
var foo=new Shuffle();
addChild(foo);
foo.doit({
           toTarget:pic,
           fromTarget:pic2,
           duration:40,
           easing:None.easeInOut});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`mode`: **String**, the direction of shuffling.. (default "left") values are:
- "left"
- "right"

Blinds

Description:
Masks the target in tiles (like blinds) using specified ordering, easing and overlap

Use example:
```
var foo=new Blinds();
addChild(foo);
foo.doit({
           toTarget:pic,
           fromTarget:pic2,
           duration:40,
           easing:None.easeInOut,
            rows:4,
           columns:4,
           overlap:0.92,
           mode:"center",
```

```
            ordering:"random"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`columns`: **int**, Number of horizontal slices to cut image (default 1)
`rows`: **int**, Number of vertical slices to cut image (default 1)
`ordering`: **String**, an ordering function to apply to the tiles, see Orderings (default "left-right")
`overlap`: **Number (0-1)**, the percentage of overlap between tiles, 0 is no overlap, 1 full overlap, 0.5 half overlap etc.. (default 0.8)
`mode`: **String**, the start point from which the masks grow (default "left"), values are:

- "right"
- "top"
- "bottom"
- "top-left"
- "bottom-right"
- "bottom-left"
- "top-right"
- "center"
- "left"

FadeTiles

Description:
Fades the target in tiles using specified ordering, easing and overlap

Use example:

```
var foo=new FadeTiles();
addChild(foo);
foo.doit({
            toTarget:pic,
            fromTarget:pic2,
            duration:40,
            easing:None.easeInOut,
             rows:4,
            columns:4,
            overlap:0.92,
            ordering:"random"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`columns`: **int**, Number of horizontal slices to cut image (default 1)
`rows`: **int**, Number of vertical slices to cut image (default 1)
`ordering`: **String**, an ordering function to apply to the tiles, see Orderings (default "left-right")
`overlap`: **Number (0-1)**, the percentage of overlap between tiles, 0 is no overlap, 1 full overlap, 0.5 half overlap etc.. (default 0.8)

FlipTiles

Description:
Flips in 3D the target in tiles using specified ordering, easing and overlap

Use example:
```
var foo=new FlipTiles();
addChild(foo);
foo.doit({
            fromTarget:pic1,
            toTarget:pic2,
            duration:40,
            easing:Bounce.easeOut,
            rows:1,
           columns:4,
           overlap:0.92,
           ordering:"random",
           spin:"vertical"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`spin`: **String**, values are "horizontal" or "vertical" , spin of flipping (default "horizontal")
`columns`: **int**, Number of horizontal slices to cut image (default 1)
`rows`: **int**, Number of vertical slices to cut image (default 1)
`ordering`: **String**, an ordering function to apply to the tiles, see Orderings (default "spiral-top-left")

`overlap`: **Number (0-1)**, the percentage of overlap between tiles, 0 is no overlap, 1 full overlap, 0.5 half overlap etc.. (default 0.8)
`background`: **Boolean**, whether a black background will be added to enhance the effect.. (default false)

FlipPage

Description:
Flips the target like it is flipping a sheet of paper

Use example:
```
var foo=new FlipPage();
addChild(foo);
foo.doit({
             toTarget:pic1,
           fromTarget:pic2,
            duration:40,
            easing:Strong.easeOut,
            mode:"diagonal-top-left-reverse"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`mode`: **String**, the start point of the flip and the direction, values are:

- "horizontal-top-left"
- "horizontal-top-right"
- "horizontal-bottom-right"
- "horizontal-bottom-left"
- "vertical-top-left"
- "vertical-top-right"
- "vertical-bottom-left"
- "vertical-bottom-right"
- "horizontal-top-left-reverse"
- "horizontal-top-right-reverse"
- "horizontal-bottom-right-reverse"
- "horizontal-bottom-left-reverse"
- "vertical-top-left-reverse"
- "vertical-top-right-reverse"
- "vertical-bottom-left-reverse"
- "vertical-bottom-right-reverse"

- "diagonal-top-right-reverse"
- "diagonal-bottom-right-reverse"
- "diagonal-bottom-left-reverse"
- "diagonal-top-left-reverse"
- "diagonal-top-right"
- "diagonal-bottom-right"
- "diagonal-bottom-left"
- "diagonal-top-left"

(default "diagonal-top-left-reverse")

FlipPage3D

Description:
Flips the target in 3D style using bending or twisting

Use example:
```
var foo=new FlipPage3D();
addChild(foo);
foo.doit({
             toTarget:pic1,
           fromTarget:pic2,
            duration:40,
            easing:Strong.easeOut,
            mode:"bend"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`mode`: **String**, the bend or twist applied (default : "bend"), values are:
- "bend"
- "twist"

ScaleTiles

Description:
Scales the target in tiles and rotates them using specified ordering, easing and overlap

Use example:
```
var foo=new ScaleTiles();
```

```
addChild(foo);
foo.doit({
            fromTarget:pic1,
           toTarget:pic2,
            duration:40,
            easing:Strong.easeOut,
            rows:6,
           columns:6,
           overlap:1,
           reverse:true,
           rotate:true,
           ordering:"columns-first"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`columns`: **int**, Number of horizontal slices to cut image (default 1)
`rows`: **int**, Number of vertical slices to cut image (default 1)
`ordering`: **String**, an ordering function to apply to the tiles, see Orderings (default "random")
`overlap`: **Number (0-1)**, the percentage of overlap between tiles, 0 is no overlap, 1 full overlap, 0.5 half overlap etc.. (default 0.5)
`rotate`: **Boolean**, whether to also rotate tiles as it scales them (default true)
`reverse`: **Boolean**, whether to apply the reverse transition,ie the tiles of fromTarget scale to zero instead of the tiles of toTrarget to scale to one (default false)

Fly

Description:
Flies in the target from specified point

Use example:

```
var foo=new Fly();
addChild(foo);
foo.doit({
           toTarget:pic1,
           fromTarget:pic0,
            duration:40,
            easing:Strong.easeOut,
           mode:"top-left"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`mode`: **String**, the start point, values are:

- "top-left"
- "top-right"
- "bottom-left"
- "bottom-right"
- "left"
- "right"
- "bottom"
- "top"

(default "top-left")

Cubes

Description:
Slices the target in cubes and rotates them using specified ordering, easing and overlap and direction

Use example:

```
var foo=new Cubes();
addChild(foo);
foo.doit({
            fromTarget:pic1,
            toTarget:pic2,
            duration:40,
            easing:Strong.easeOut,
            slices:3,
            ordering:"columns-first",
           slicing:"horizontal",
           overlap:0.9,
           direction:"left"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`slices`: **int**, Number of slices to cut image (default 1)
`ordering`: **String**, an ordering function to apply to the tiles, see Orderings (default "columns-first")
`overlap`: **Number (0-1)**, the percentage of overlap between tiles, 0 is no overlap, 1 full overlap, 0.5 half overlap etc.. (default 0.5)
`slicing`: **String**, values are "horizontal", "vertical", whether to create horizontal or vetical slices (default "horizontal")
`direction`: **String**, direction to rotate cubelets to, values are : "left", "right", "up", "down", default ("right")
`direction_mode`: **String**, whether the direction alternates or is random or constant, values are : "none", "alternate", "random", default ("none")
`background`: **Boolean**, whether a black background will be added to enhance the effect.. (default false)

Pixelate

Description:
Zooms the target to pixel size and then back to new target using specified easing

Use example:
```
var foo=new Pixelate();
addChild(foo);
foo.doit({
            fromTarget:s9,
            toTarget:st10,
            duration:40,
            easing:None.easeOut,
            geometric:true});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`geometric`: **Boolean**, whether the scaling is geometrical or linear (default true)

Clock

Description:
Reveals the target in a clock like manner using specified easing

Use example:
```
var foo=new Clock();
addChild(foo);
foo.doit({
            toTarget:st3,
           fromTarget:st2,
            duration:40,
            easing:Strong.easeOut,
            segments:4});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`segments`: **int**, the number of segments to divide clock (default 1)
`start_angle`: **Number**, the start angle in degrees (default 0)
`clockwise`: **Boolean**, whether the direction is clockwise or not (default false)

GradientSweep

Description:
Reveals the target using gradient alpha mask from specified point

Use example:
```
var foo=new GradientSweep();
addChild(foo);
foo.doit({
            fromTarget:pic1,
             toTarget:pic2,
             duration:40,
             easing:None.easeInOut,
             mode:"bottom-right"});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`mode`: **String**, the start point and the direction of the sweep values are:

- "top-left"
- "top-right"
- "bottom-left"
- "bottom-right"
- "left"
- "right"
- "top"
- "bottom"
- "radial"

(default "top-left")

GradientMasks

Description:
Reveals the target using gradient alpha masks either horizontally or vertically with specified ordering, easing and overlap

Use example:

```
var foo=new GradientMasks();
addChild(foo);
foo.doit({
            fromTarget:pic1,
             toTarget:pic2,
             duration:60,
             easing:Back.easeOut,
             slices:6,
           mode:"vertical",
           overlap:0.8,
           ordering:"random",
           alternate:true});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`slices`: **int**, number of masks (default 3)
`alternate`: **Boolean**, whether the masks are moving alternately (default true)
`mode`: **String**, either "horizontal" or "vertical" (default "horizontal")
`ordering`: **String**, an ordering function to apply to the tiles, see Orderings (default "random")
`overlap`: **Number (0-1)**, the percentage of overlap between tiles, 0 is no overlap, 1 full overlap, 0.5 half overlap etc.. (default 0.5)

DreamWave

Description:
Fades in the target in a dream-like waving manner

Use example:
```
var foo=new DreamWave();
addChild(foo);
foo.doit({
            fromTarget:pic1,
             toTarget:pic2,
             duration:60,
             easing:Back.easeOut});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition
`duration`: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
`easing`: **Function**, easing function to apply to transition

`speedx`: **Number**, speed of waving in x-axis (default 20)
`speedy`: **Number**, speed of waving in y-axis (default 20)
`maxx`: **Number**, maximum distortion in x-axis (default 20)
`maxy`: **Number**, maximum distortion in y-axis (default 20)

Ripples

Description:
Fades in the target like it is inside rippled water

Use example:
```
var foo=new Ripples();
addChild(foo);
foo.doit({
            fromTarget:pic1,
             toTarget:pic2,
             duration:40,
             easing:None.easeOut});
```

Parameters:
`fromTarget`: **DisplayObject**, DisplayObject at start of transition
`toTarget`: **DisplayObject**, DisplayObject at end of transition

duration: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
easing: **Function**, easing function to apply to transition

maxs: **Number**, maximum distortion of ripple (default 60)
dens: **Number (0-10)**, density of ripples (default 2)

Slices

Description:
Slices the target and moves in or out the slices in specified angle using specified ordering, easing and overlap

Use example:
```
var foo=new Slices();
addChild(foo);
foo.doit({
             toTarget:st4,
           fromTarget:st3,
            duration:80,
            easing:Elastic.easeOut,
            overlap:0.97,
           alternate:true,
           type:"in",
           area:100,
           ordering:"random",
           angle:45});
```

Parameters:
fromTarget: **DisplayObject**, DisplayObject at start of transition
toTarget: **DisplayObject**, DisplayObject at end of transition
duration: **Number**, duration of the whole transition in seconds or frames (depending on useFrames or useFramesGlobal parameter)
easing: **Function**, easing function to apply to transition

type: **String**, either "in" or "out", whether the transition is in-transition or out-transition (default "in")
area: **Number**, the width or height in pixels of each slice (default 10)
angle: **Number**, the angle in degrees at which the slicing is done (default 45)
alternate: **Boolean**, whether the masks are moving alternately (default true)
ordering: **String**, an ordering function to apply to the tiles, see Orderings (default "random")
overlap: **Number (0-1)**, the percentage of overlap between tiles, 0 is no overlap, 1 full overlap, 0.5 half overlap etc.. (default 0.8)